УДК 517.957]:(005.336.1-042.2:004.424.44:514.112)

Афанасьєва Д. С., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Потапова Н. А., *канд. екон. наук, доцент, доцент кафедри інформаційних технологій*

ПОРІВНЯННЯ ЕФЕКТИВНОСТІ МЕТОДІВ ДИХОТОМІЇ ТА ХОРД У РОЗВ'ЯЗАННІ НЕЛІНІЙНИХ РІВНЯНЬ

Донецький національний університет імені Василя Стуса, м. Вінниця

В сучасній математичній та інженерній практиці розв'язання нелінійних рівнянь відіграє важливу роль у вирішенні різноманітних завдань, їх ефективне розв'язання має велике значення для розвитку науки та технологій. Одними з провідних методів розв'язання таких рівнянь є методи дихотомії та хорд. Проте для успішного застосування цих методів у практичних задачах необхідно детально оцінити їх ефективність та області застосування.

Для знаходження кореня рівняння $f(x) = 0$ з деякою точністю ε на відрізку $[a, b]$, де функція є неперервною та має значення різних знаків на кінцях a та b, можна використати метод дихотомії. Процес виконання методу полягає у такому: спочатку визначається середина відрізка $[a,b]$ (точка c). Якщо корінь рівняння співпадає зі значенням c, то обрахунки зупиняються, і результат обчислення отриманий. В іншому випадку корінь знаходиться на відрізку $[a, c]$ чи $[b, c]$, тоді значення для наступних ітерацій визначаються за допомогою перевірки умов $f(a) * f(c) < 0$ та $f(b) * f(c) < 0$. Далі алгоритм повторюється для відрізка, що містить розв'язок, доти, доки довжина відрізка не стане меншою від заданої похибки [1]. Приклад програмної реалізації методу дихотомії з використанням мови програмування Python показано на рис. 1.

```
def bisection_method(a, b, tol=0.5e-6, max_iter=1000):
    if func(a) * func(b) >= 0:
        print("Метод дихотомії не може бути використаний, оскільки "
              "на кінцях відрізка функція має однаковий знак.")
        return None
    iter_count = 0
    while (b - a) / 2.0 > tol:
        c = (a + b) / 2.0
        if func(c) == 0:
            return c
        elif func(c) < 0:
            a = c
        else:
            b = c
        iter_count += 1
        if iter_count > max_iter:
            print("Досягнуто максимальну кількість ітерацій.")
            return None
    print(f"Кількість ітерацій методу дихотомії: {iter_count}")
    print("Розв'язок методом дихотомії:", round(((a + b) / 2.0), 6))
```

Рис. 1. Програмна реалізація методу дихотомії

У конструкції while програмного коду показано принцип визначення a та b для кожної ітерації. Якщо $f(c) < 0$, то відрізок набуває значень $[c, b]$, якщо $f(c) > 0$ – $[a, c]$.

Використання методу хорд для знаходження кореня рівняння $f(x) = 0$ на відрізку $[a, b]$, де функція є неперервною, та $f(a) * f(b) < 0$, з деякою точністю ε базується на побудові певного числа прямих, що перетинають вісь x та проходять через дві початкові точки $f(x)$. Алгоритм реалізації цього методу включає пошук похідних першого та другого порядків для функції $f(x)$ на кінцях відрізка, та перевірку, чи знак похідних на них не змінюється. Після цього обирається значення початкового наближення, як будь-яка точка на відрізку, де виконується умова $f(x) * f''(x) < 0$. Далі визначаємо нерухомий кінець c, як будь-яку точку на відрізку, де виконується умова $f(\mathrm{c}) * f''(\mathrm{c}) > 0$. Значення x для першої ітерації буде рівне початковому наближенню, а для наступних ітерацій буде обчислене за формулою (1) [2]:

$$x_{k+1} = x_k - \frac{f(x_k)}{f(x_k) - f(c)}(x_k - c). \quad (1)$$

Продовжувати ітерації треба, поки абсолютне значення різниці між поточним та попереднім наближеннями менше за задану точність. Приклад програмної реалізації методу хорд з використанням мови програмування Python, де похідні першого та другого порядків визначаються попередньо заданими функціями first_derivative() та second_derivative(), показано на (рис. 2):

```
def chord_method(a, b, epsilon=0.5e-6, max_iter=1000):
        if first_derivative(a) * second_derivative(a) < 0 or \
            first_derivative(b) * second_derivative(b) < 0:
        print("На кінцях відрізка змінюється знак похідної")
        return None

    x0 = a
    while True:
        if func(x0) * second_derivative(x0) < 0: break
        else: x0 += 0.1

    c = b
    while True:
        if func(c) * second_derivative(c) > 0: break
        else: c -= 0.1

    iter_count = 0
    fc = func(c)
    while True:
        fx0 = func(x0)
        x = x0 - ((fx0 / (fx0 - fc))*(x0 - c))
        if abs(x - x0) < epsilon or iter_count >= max_iter: break

        x0 = x
        iter_count += 1

    print(f"Кількість ітерацій: {iter_count}")
    print("Розв'язок методом хорд:", round(x, 6))
```

Рис. 2. Програмна реалізація методу хорд

Після проведення обчислень за допомогою методів дихотомії та хорд можна здійснити порівняльний аналіз їх ефективності згідно з отриманими результатами:

1. Для функції, що була обрана для проведення дослідження, метод хорд є ефективнішим з погляду швидкості збіжності. Адже з отриманих результатів виконання програм можна побачити, що метод хорд потребував лише 8 ітерацій, порівняно з 20 ітераціями методу дихотомії (рис. 3).

```
Кількість ітерацій методу дихотомії: 20
Розв'язок методом дихотомії: 0.297786

Кількість ітерацій: 8
Розв'язок методом хорд: 0.297786
```

Рис. 3. Результати виконання програм

2. Зважаючи на те, що результати методу хорд були здебільшого залежні від зміни знака похідної, метод хорд може виявитися менш стійким до нульових похідних, ніж метод дихотомії.

3. Метод дихотомії не потребує визначення знаків похідних та зазвичай обмежується вибором кінців інтервалу. Натомість метод хорд може вимагати більше обчислень для вибору початкових значень, що може збільшити обчислювальну складність.

Внаслідок дослідження було встановлено, що для цього випадку метод хорд виявився більш ефективним, порівняно з методом дихотомії. Адже у практичних обчисленнях метод хорд продемонстрував меншу кількість ітерацій, порівняно з методом хорд, що свідчить про його більшу ефективність. Проте метод дихотомії у деяких випадках може бути більш простим у застосуванні, оскільки не вимагає знання знаків похідних та обмежується вибором кінців інтервалу. Отже, вибір методу залежить від конкретної задачі та вимог до швидкості і простоти обчислень.

Список використаних джерел

1. Комп’ютерне моделювання систем та процесів. Методи обчислень. Частина 1: навчальний посібник / Р. Н. Квєтний, І. В. Богач, О. Р. Бойко, О. Ю. Софина, О. М. Шушура. Вінниця: ВНТУ, 2012. 193 с. URL: http://kist.ntu.edu.ua/textPhD/kmsp.pdf

2. Чисельні методи: навчальний посібник / Л. О. Волонтир, О. В. Зелінська, Н. А. Потапова, І. А. Чіков. Вінницький національний аграрний університет. Вінниця: ВНАУ, 2020. 322 с.

3. Верещак Р. Знаходження наближеного розв’язку нелінійного алгебраїчного рівняння методом хорд. *www.mathros.net.ua: сайт для студентів спеціальності інформатика.* 2012. URL: https://www.mathros.net.ua/znahodzhennja-nablyzhenogo-rozvjazku-nelinijnogo-algebraichnogo-rivnjannja-metodom-hord.html (дата звернення: 02.05.2024).

4. Smit A. Program for Bisection Method. *geeksforgeeks.org.* 2022. URL: https://www.geeksforgeeks.org/program-for-bisection-method/ (дата звернення: 02.05.2024).